KLE Technological University

Creating Value
Leveraging Knowledge

INDUSTRIAL PROJECT REPORT

Internship project report on

Design and Validation of Safety IP on SoC

submitted in partial fulfilment of the
Requirements for the award of

Bachelor of Engineering
in
School of Electronics and Communication Engineering

Carried out at

Bosch Global Software Technologies Private Limited (BGSW)

Submitted By:
Suraj Baddi
01FE19BEC268

Under the guidance of

Prof. Rohit Kalyani
College Guide
KLE Technological University

Mr. Sah Manish Kumar
Industry Guide
Bosch Global Software Tech

SUBMITTED TO:
School of Electronics and Communication Engineering
KLE TECHNOLOGICAL UNIVERSITY
Hubballi

K.L.E SOCIETY'S
KLE Technological University,
HUBBALLI-580031
2022-2023

SCHOOL OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to attest to the fact that the project titled **" Design and Implementation of Functionally Safe Automotive SoC "** is a bonafide work carried out by the student **" Suraj Baddi [01FE19BEC268] "** studying in final year under Industrial internship training at **Bosch Global Software Technologies** in Bangalore from **20/01/2023 to 31/05/2023** in partial fulfillment for the award for Bachelor of Engineering in Electronics and Communication in the School of Electronics and Communication Engineering of KLE Technological University, Hubballi for the academic year 2022-2023.

Prof. Rohit Kalyani	**Dr. Nalini C. Iyer**	**Dr. Basavaraj S. Anami**
Guide	**Head of School**	**Registrar**

External Viva:

Name of Examiners **Signature with date**

1.

2.

ACKNOWLEDGMENT

The happiness and exhilaration that accompany the successful completion of any task would be insufficient if I did not acknowledge a number of people whose expert advice and encouragement assisted me in the successful completion of this report work. I take this opportunity to thank Dr. Ashok Shettar, Vice-chancellor, KLE Technological University and to Dr. Prakash Tewari, Principal, KLE Technological University, Hubballi. I also take this opportunity to thank Dr. Nalini Iyer, Head of School, School of Electronics and Communication Engineering for providing the opportunity of industry internship at Bosch Global Software Technologies which nurtured our practical skills contributing to the success of our project.

I sincerely thank my guide Mr. Sah Manish Kumar (MS/EEH21-PS), Bosch Global Software Technologies Private Limited (BGSW) for his guidance, inspiration and wholehearted cooperation during the course of completion. I sincerely thank Mr. Rudrappa Gouda (MS/EEH22-PS) and Mr. Prakash Shetty (MS/EEH2-PS), for the valuable guidance and advice who inspired me greatly to work in this project. Their willingness to motivate, contributed tremendously to the project.

I would also like to thank Prof. Rohit Kalyani, School of Electronics and Communication Engineering for his support and encouragement.

- Suraj Baddi (01FE19BEC268)

COMPANY BACKGROUND

Bosch Global Software Technologies (BGSW) is the software development unit of Bosch, a leading global supplier of technology and services. BGSW was established in 2011 with the aim of developing cutting-edge software solutions that enable Bosch to stay ahead of the curve in the rapidly evolving technology landscape.With a workforce of over 15,000 associates spread across 30 countries, operates in a wide range of industries such as automotive, healthcare, energy, and building technology.

BGSW's software development expertise encompasses a wide range of technologies, including artificial intelligence, machine learning, cloud computing, and the Internet of Things (IoT). The company's software solutions cover a broad spectrum of applications, from automotive and industrial automation to smart homes and buildings, and from healthcare and energy management to connected mobility.

BGSW's commitment to quality is reflected in its adherence to international software development standards such as Automotive SPICE, ISO 26262, and IEC 61508. The company also places a strong emphasis on data privacy and security, ensuring that its software solutions are designed to meet the highest standards of cybersecurity.

ABSTRACT

The rapid advancements in automotive technology have led to the integration of sophisticated electronic systems in modern vehicles. To ensure reliable and secure operation of these systems, functional safety has become a paramount concern for the automotive industry. This paper presents the design and implementation of a functionally safe System-on-Chip (SoC) specifically tailored for automotive applications.

The proposed SoC architecture addresses the challenges of functional safety by incorporating robust safety mechanisms at various levels of the design. The safety features include fault detection, error correction, redundancy, and comprehensive self-diagnostic capabilities. The design leverages state-of-the-art safety standards, such as ISO 26262, to ensure compliance with the stringent safety requirements of the automotive domain.

A key aspect of the SoC design is the integration of dedicated safety modules, such as safety controllers, fail-safe memory, and built-in self-test circuits. These modules work synergistically to monitor and mitigate potential faults within the SoC, ensuring that the system continues to operate reliably even in the presence of faults. Furthermore, the SoC architecture incorporates safety mechanisms at the interface level to enable safe communication between the SoC and external automotive subsystems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The automotive industry is constantly evolving with the rapid integration of electronic systems into vehicles. These electronic systems are becoming increasingly complex and interconnected, ADAS , infotainment systems, and safety-critical functions such as braking and steering. While these systems bring numerous benefits to drivers and passengers, they also pose a significant safety challenge. The risk of hardware failure in these systems could result in serious accidents and harm to people. One of the primary solutions to mitigate this risk is the use of functionally safe automotive system-on-chip (SoC) designs. As shown in Fig. 1.1, in-vehicle functional safety analysis is split into three levels of abstraction.

1. Vehicle level .
2. Component/ECU level and
3. Chipset/SoC level.

In this case study, we will focus on the design of the functional safety concept for the Automotive System on Chip (SoC) that is specifically intended to meet the ISO 26262 criteria.

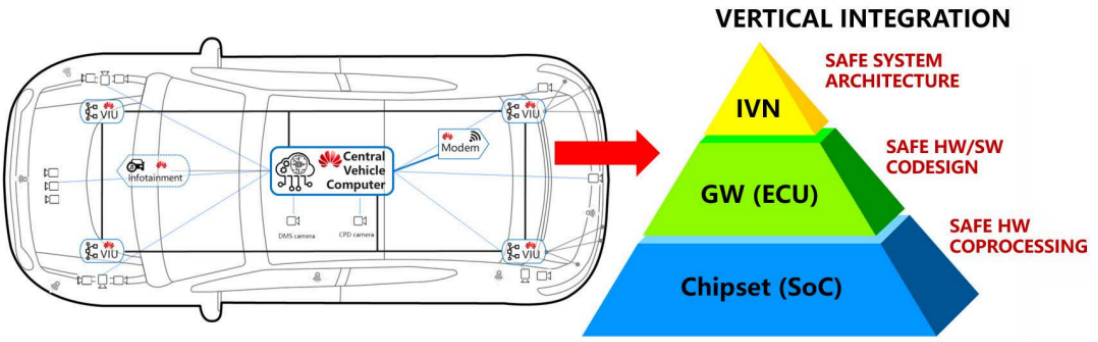

Figure 1.1: Three levels of abstraction used in functional safety analysis.

Functionally safe SoCs are designed to meet stringent safety requirements, ensuring reliable and safe operation even in the event of hardware failures. These SoCs are critical components in modern vehicles, and their reliability and safety are of utmost importance. The design and implementation of a functionally safe automotive SoC requires a deep understanding of the safety requirements and standards such as ISO 26262. Designers must employ a variety of safety mechanisms and design techniques to ensure reliable and safe operation. For instance, they can use redundancy, error-correction codes, and fail-safe mechanisms to ensure the system can detect and respond to hardware failures.

This paper presents the design and implementation of a functionally safe automotive SoC. The SoC was designed using advanced safety mechanisms and design techniques to meet the stringent safety requirements of modern vehicles. The paper will provide a detailed description of the design process, the selection of safety mechanisms, and the integration of these mechanisms into the SoC design.

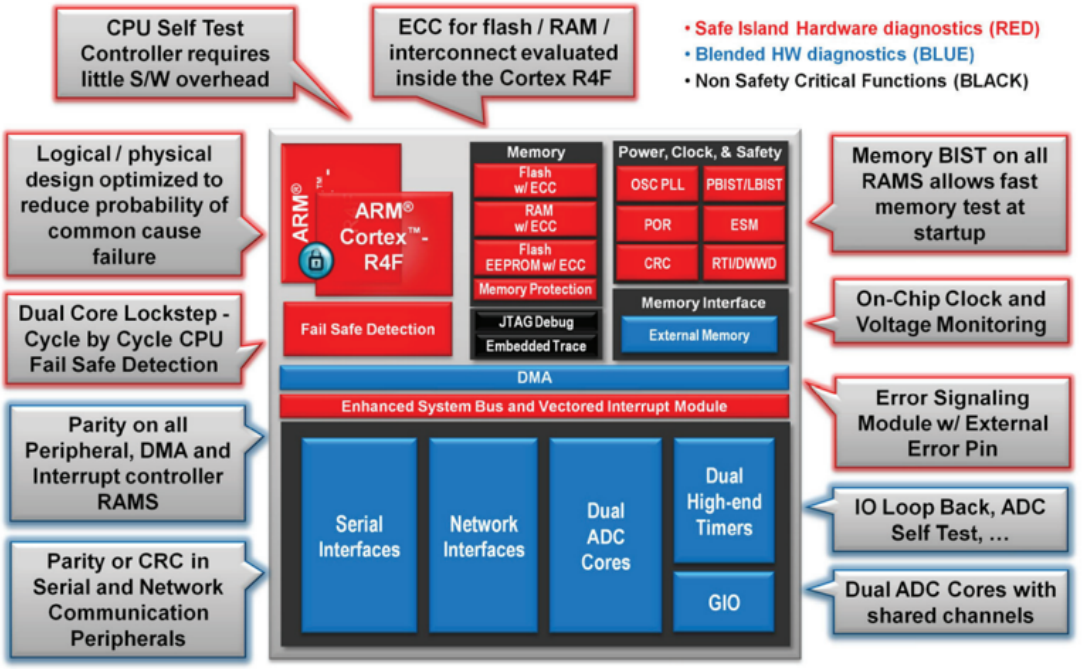

Figure 1.2: Design of SoC with Safety Features

The paper will also discuss the testing and verification of the SoC done using UDE tool. Testing and verification are critical to ensuring that the SoC meets the safety requirements and operates as intended in various operating conditions and failure scenarios. The UDE (Universal Debug Engine) serves as a valuable tool for evaluating and ensuring the safety of the any micro controller hardware. It provides users with a comprehensive view of the safety tests conducted on the device. Users can access detailed information regarding the test procedures, outcomes, and any alarms triggered during the tests. The paper highlights the critical role that functionally safe SoCs play in ensuring the safe and reliable operation of modern vehicles and the need for rigorous design, testing, and verification processes to ensure their effectiveness.

1.1 Motivation

As vehicles become more reliant on electronic systems, ensuring their safety has become a significant challenge for the automotive industry. The implementation of safety standards such as ISO 26262 highlights the need for reliable and safe electronic systems. Functionally safe automotive system-on-chip (SoC) designs are critical to meeting these safety requirements and mitigating the risks associated with hardware failures. This paper provides a detailed description of the design and implementation process of a functionally safe automotive SoC, including the selection of safety mechanisms, testing, and verification. The paper aims to contribute to the development of safe and reliable electronic systems in modern vehicles and prevent safety-critical incidents. The findings will be of interest to automotive manufacturers, electronic system designers, and anyone involved in the development of safety-critical systems.

1.2 Objectives

The following are the key objectives which were focused:

1. Understanding the Need for Functional Safety and the ISO26262 standard.
2. To research the safety aspects particular to processors and System-on-Chip design methods.
3. To study about different faults and the Fault Management Module: FCCU (Fault Collection and Control Unit).
4. To learn about UDE (Universal Debug Engine), a tool used for system analysis, testing, and debugging.
5. Finally, to perform various Safety tests on IFX device 4 and analyse the generated alarms using UDE.

1.3 Literature Survey

The paper proposes a methodology for designing system-on-chips (SoCs) with embedded processors that can achieve high reliability. The proposed methodology includes several key design techniques such as redundancy, error correction codes, and fault-tolerant architectures, which can help mitigate the effects of hardware and software faults. It also discusses the challenges associated with designing high-reliability systems, such as the need for fault tolerance, the need to minimize power consumption, and the need to maximize performance. The authors propose a design methodology that includes a combination of hardware and software techniques to achieve these goals. Overall, the paper provides valuable insights into the challenges and techniques involved in designing high-reliability SoCs with embedded processors. [10]

The paper is designed to meet the safety requirements of autonomous driving applications. The proposed SoC is based on a multi-core processor architecture and includes a variety of hardware and software features to ensure functional safety. It also discusses the challenges associated with designing SoCs for autonomous driving, including the need to meet stringent safety requirements, the need for real-time processing, and the need for low power consumption. The authors propose a design methodology that includes a combination of hardware and software techniques to achieve these goals. The paper also presents a case study of the SoC design for an autonomous driving application. The authors describe the design process and the key features of the SoC, including safety mechanisms such as hardware-based safety monitors, redundancy, and fault detection and correction. [9].

The authors discuss various functional safety methodologies that are currently used in the industry and provide an overview of the challenges associated with the implementation of these methodologies. The paper begins by discussing the importance of functional safety in the automotive industry and the need for methodologies that can ensure the safety of complex systems. The authors then provide an overview of the various functional safety standards that are currently used in the industry, including ISO 26262 and the Automotive Safety Integrity Level (ASIL) classification system. The paper then delves into the different methodologies used to achieve functional safety in automotive applications. They also discuss the use of safety measures such as redundancy, diversity, and diagnostics, as well as the use of safety mechanisms such as safe state management and safe stop. Finally, the paper concludes with a discussion of the future of functional safety in the automotive industry. The authors suggest that the industry will continue to focus on functional safety as a means of ensuring the safety of complex systems, and that new methodologies and technologies will be developed to address the challenges associated with the implementation of these methodologies. [5]

The paper discusses the features and benefits of the Hercules line of microcontrollers designed by Texas Instruments for use in safety-critical applications. The paper starts by highlighting the importance of safety-critical systems in various industries such as automotive, medical, and industrial control, and the need for reliable and robust microcontrollers to power these systems. The paper then goes on to detail the various features of the Hercules line of microcontrollers that make them suitable for safety-critical applications. These features include high-performance processing capabilities, real-time monitoring and diagnostics, robust memory protection, and built-in safety mechanisms such as watchdog timers and error-correcting code (ECC) memory. The paper also highlights the various safety standards and certifications that the Hercules microcontrollers meet, such as ISO 26262, IEC 61508, and SIL3, which are required for safety-critical applications in different industries. Furthermore, the paper provides real-world examples of how the Hercules microcontrollers have been used in various safety-critical applications such as airbag control systems, medical infusion pumps, and industrial automation systems. [2]

A method for testing error-correcting code (ECC) protected memories using an on-chip memory built-in self-test (MBIST) device. ECC is a technique used to detect and correct errors in memory systems, and it is widely used in safety-critical applications where data integrity is critical. The paper begins by discussing the importance of ECC in memory systems and the need for reliable testing methods to ensure the effectiveness of ECC. The authors then propose a novel MBIST device that can test ECC-protected memories by injecting errors into the memory system and checking the ECC error correction mechanism's response. The paper provides a detailed description of the MBIST device's architecture, which consists of multiple test patterns that target different types of errors in the memory system. The device is designed to work with various types of ECC schemes, such as single-bit error correction (SEC) and double-bit error detection (DED) schemes. The authors also present simulation results that demonstrate the effectiveness of the proposed MBIST device in detecting and correcting errors in ECC-protected memories. The results show that the device can detect and correct all single-bit errors and most double-bit errors, making it a reliable and efficient method for testing ECC-protected memories. [3].

The application of Built-In Self-Test (BIST) techniques to the MPC5744P microcontroller. BIST is a testing technique that allows integrated circuits to be tested during operation, enabling a more efficient and cost-effective way of testing. The paper begins by introducing the MPC5744P microcontroller, which is used in safety-critical applications such as automotive control systems. The authors then discuss the importance of testing microcontrollers in safety-critical applications and the limitations of traditional testing techniques. They propose the use of BIST as a more efficient and reliable testing technique for the MPC5744P. The authors provide a detailed description of the BIST techniques used in the MPC5744P, including the logic BIST, memory BIST, and analog BIST. They also describe the implementation of the BIST

techniques using the microcontroller's on-chip resources, such as timers, counters, and analog-to-digital converters. The paper also presents the results of experiments conducted to test the effectiveness of the BIST techniques in the MPC5744P. The results show that the BIST techniques can detect and diagnose faults in the microcontroller, including those caused by aging, manufacturing defects, and environmental factors. [8]

1.4 Problem Statement

" Design and Implementation of Functionally Safe Automotive System-on-Chip "

1.5 Application in Societal Context

The application of such technology has the potential to significantly reduce the number of accidents and fatalities caused by system failures in vehicles. The implementation of this technology will also enhance the trust of consumers in the safety of automotive systems, thereby increasing the adoption of new technologies. Moreover, the design and implementation of functionally safe automotive SoCs can also have significant economic implications. Automotive accidents can result in significant economic losses, including medical costs, legal expenses, and vehicle damage. By reducing the number of accidents caused by system failures in vehicles, the implementation of this technology can significantly reduce these costs, thereby benefiting both individuals and society as a whole.

1.6 Organization of the Report

We provide a brief description of the introduction, and motivation for opting for this topic in Chapter 1: the main objectives concerned for this project, literature survey, problem statement which is selected by the need of current requirement, Applications in Societal Context, which gives a brief description of the usage of this topic in society. Chapter 2: This Chapter explains about all the safety mechanisms that were designed to form a Safety Island and also about the faults that can occur. Chapter 3: This Chapter details the software and hardware that were used and also discusses about some safety tests that were run. Chapter 4: This is the project's conclusion, including a discussion of the project's future scope, such as what tools and techniques can be applied in other areas and how they might be used more effectively.

Chapter 2

Design of Safety Island

In this chapter, we will look at the Safety SoC design, which incorporates hardware-based safety features to form a "Safe Island" from which faults in the rest of the system can be detected.

2.1 Dual Core Lockstep Architecture

The cores that run inside the CPU of a SoC are divided into two domains: Application domain and Real-time domain as shown in Fig 2.1. The cores in the Application domain conduct tasks that consider safety as secondary. Cores in the Real-time domain, on the other hand, perform safety-critical operations. Therefore, cores running in real-time domain places a strong emphasis on safety, as it is commonly found in safety-critical applications where failures can lead to serious consequences, such as in automotive, aerospace, or medical systems.

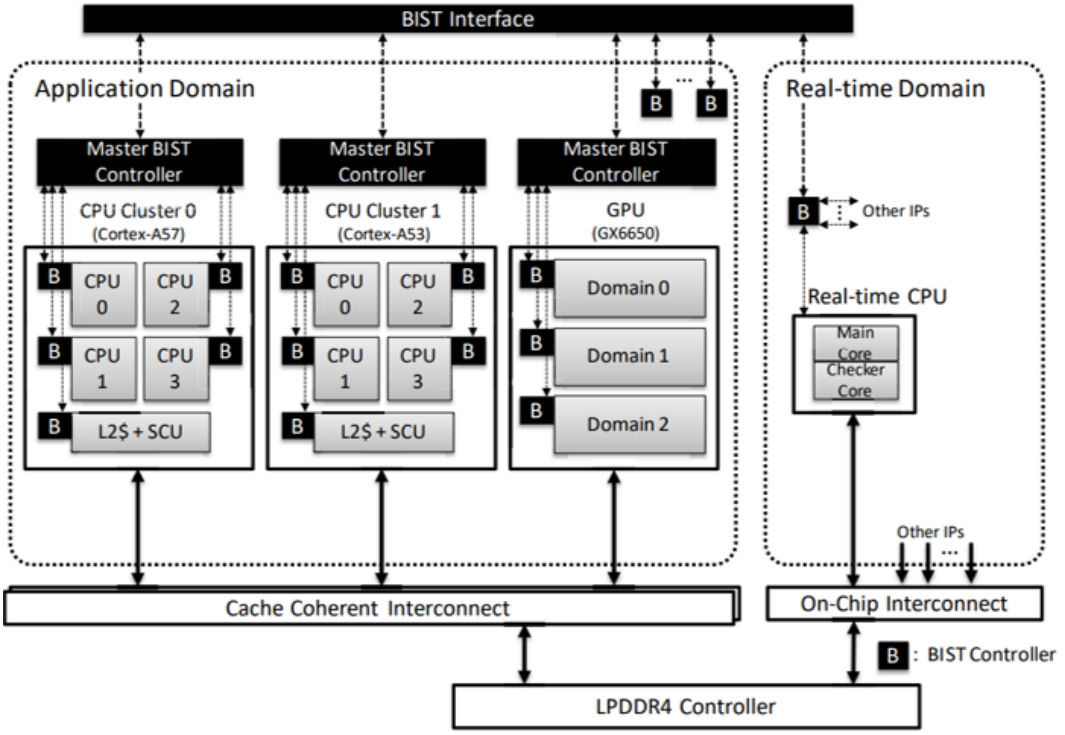

Figure 2.1: Block diagram of SoC having two clusters of quad-core application CPUs

Therefore the core running in real-time domain operates in Lockstep mode which consists of a Master and Checker core as shown in Fig 2.2. The checker core executes the same instructions as the master core, but it is 2 clock cycles out of phase. The LCL (Lockstep Comparator Logic) monitors the result of the comparison and flags an error if the state of the monitored nodes differ. A delayed lockstep technique ensures safety for the Cores, Interrupt Controllers, DMA, and their closely associated periphery. Each replicated set of elements creates a channel (for example, the Main channel and the Checker channel).

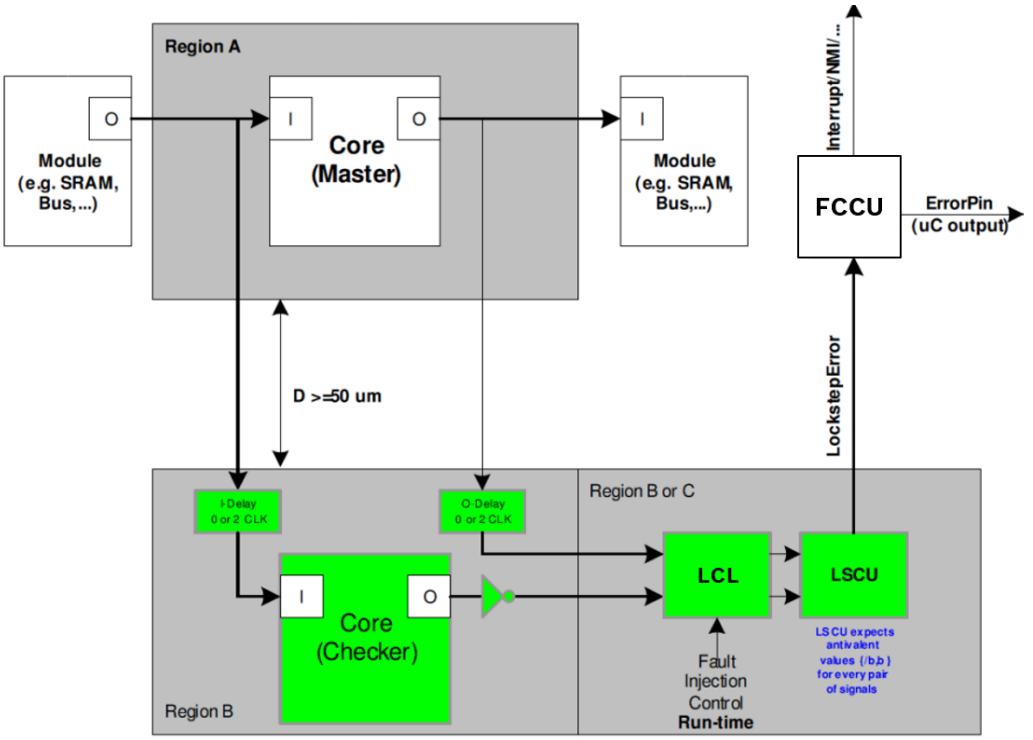

Figure 2.2: Lockstep Architecture: Delayed Lockstep Execution

2.2 Hardware Built-in-Self-Test (BIST)

The cores should perform a self-test after power-on to ensure that the checker hardware is valid/properly working after power-on. As a result, Hardware BIST is used as a safety device to detect faults. When the system is powered on, the hardware BIST runs a power-on self-test to detect latent faults. Hardware BIST is a vital component of RuntimeTEST for detecting single point problems during runtime. In general, the self-test is performed on the digital logic (called LBIST) and embedded memories (called MBIST) with sufficient coverage to achieve the system's specified Safety Integrity Level (SIL). BIST methods of verifying random logic fall into two broad types.

- **Online BIST:** Performed when the functional circuit is in use, such as during normal operation.
- **Offline BIST:** When the functioning circuitry is not in normal mode, such as during the engine's power-on reset.

2.2.1 Memory BIST and Logic BIST

Memory BIST is used to test the memory elements (such as RAM, ROM, and cache) of an IC. The primary goal of Memory BIST is to detect and identify faults or defects in the memory components, such as stuck-at faults, transition faults, or coupling faults. By using built-in self-test circuitry integrated within the memory itself. It is particularly useful for detecting faults in memories that are embedded within larger systems, as it allows for localized testing without the need for external test interfaces.

Logic BIST, on the other hand, is used to test the non-memory logic elements (such as flip-flops, multiplexers, and combinational logic) of an IC. The primary function of a logic BIST is to verify the correct functioning of the logical operations performed by the IC and detect any faults or defects such as stuck-at faults, bridging faults, delay faults, or other types of faults that can occur during the operation of the chip.

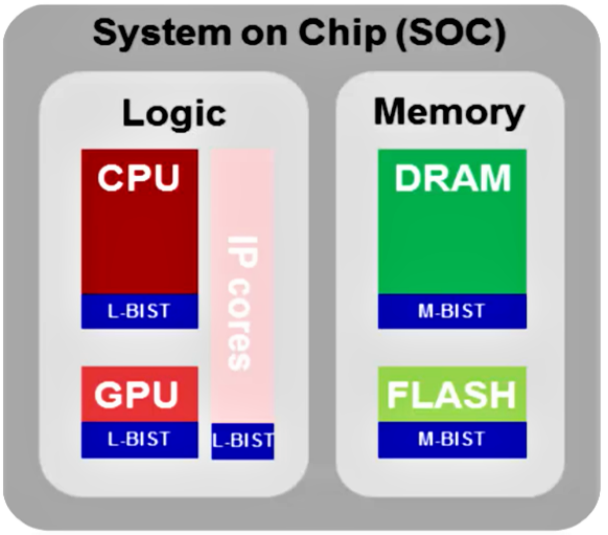

Figure 2.3: BIST Classification

2.2.2 BIST Architecture

A BIST circuit basically consists of four parts:

- **TPG (test/hardware pattern generator) :** It is used to automatically generate test vectors and pour them into the input pins of CUT (circuit under test).
- **ORA (output response analyzer/compactor) :** The output of the circuit to be tested is compressed and compared to determine whether there is an error in the circuit.
- **BIST/Test Controller :** Controls when and what data is applied to the CUT, control the clock of the circuit under test and determine when to read the expected response.
- **CUT (Circuit Under Test):** Refers to the specific portion of the integrated circuit (IC) that is being tested using the BIST architecture. It can range from simple logic gates or flip-flops to complex processors, memory arrays, or system-on-chip (SoC) components.

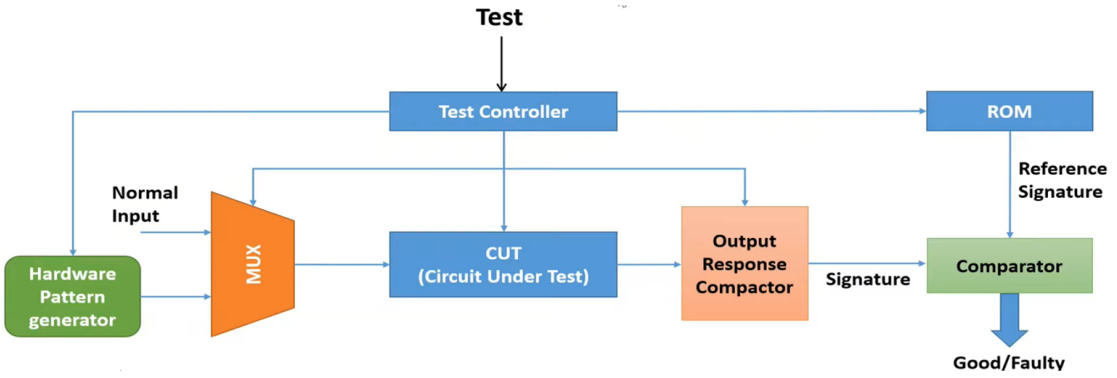

Figure 2.4: BIST Architecture

2.3 Self-Test Control Unit (STCU)

STCU is a module found in certain microcontrollers, specifically in the SPC58 series of microcontrollers developed by STMicroelectronics. It is a hardware module that may be programmed to control the self-test sequence used in either offline or online conditions, or in both. The hardware of the device can control its memory (SRAM or ROM) and logic (LBIST) built-in self-test (MBIST) blocks.

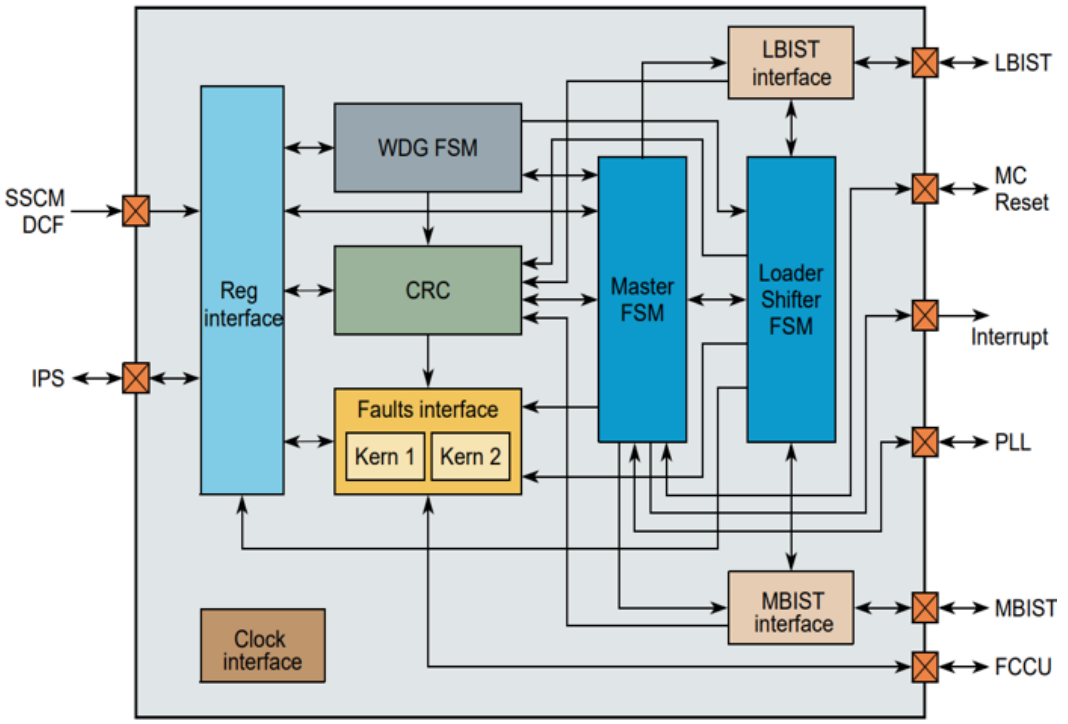

Figure 2.5: Block Diagram of STCU

The BIST architecture includes a Master FSM for test control, a Loader Shifter FSM for parameter/test result interaction, a WDG FSM for timeout monitoring and interrupt signals, a

Clock Block for clock management, a Fault interface for collecting error conditions, a CRC for self-checking, and an FCCU signal for classifying signals. Together, these components facilitate efficient testing, error detection, and fault management within the integrated circuit.

2.4 Error Correcting Codes (ECC)

During read and write operations on memory, which occur frequently, there is a possibility of digital bits experiencing flips, changing from a one (1) to a zero (0) or vice versa. These bit flips can result from various factors such as component aging, external attacks, or even cosmic rays. In safety-critical scenarios, such as those involving critical data paths, bit flips can pose significant risks. Therefore, it is crucial for safety-critical memory systems to incorporate error correctable codes (ECCs). The occurrence of bit flips and the efficacy of error correction depend on several factors, including the specific ECC algorithm utilized and the number of bit flips. Among the popular error-correctable codes, the Hamming code stands out as it can detect a maximum of two errors and correct single-bit errors, while remaining relatively straightforward to implement.

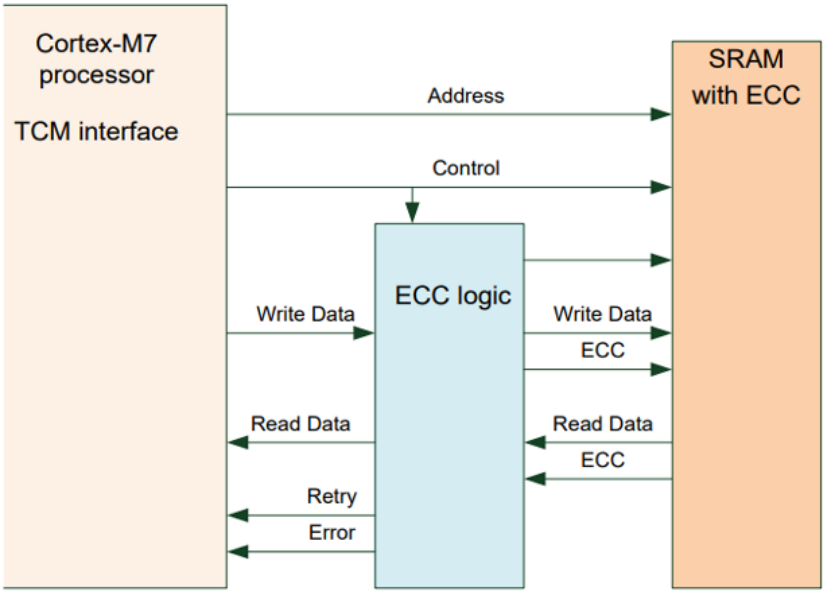

Figure 2.6: ECC Working

During a WRITE operation, checksum bits, also known as ECC bits, are generated using an ECC encoder for the input data word. These ECC bits are stored in the SRAM alongside the data word itself, providing additional protection. When a READ operation occurs, an ECC decoder compares the read data word along with the ECC bits to the anticipated ECC value. If any random hardware failure or soft error causes corruption in the SRAM-stored data word, a mismatch will occur compared to the expected ECC value. In such cases, an error and alarm are promptly notified. ECC are employed to ensure end-to-end protection by including:

- SEC/DED (single bit error correction and double bit error detection) is used for core, local memory, and peripheral RAM.
- SEC/DEC/TED (single bit error correction, double bit error correction and triple error bit detection) is used for flash memory.

- The caches in the cores are protected by EDC (Error Detection Code), only detecting, but not correcting errors.

2.5 Memory Error Management Unit (MEMU)

A Memory Error Management Unit (MEMU) is a component of a SoC/controller that is responsible for detecting and correcting errors that occur in memory. The MEMU monitors memory accesses and performs error detection and correction techniques to identify and rectify memory errors. There are different types of memory errors that can occur, such as single-bit errors, multi-bit errors, and burst errors. The MEMU typically employs error detection codes (such as parity or cyclic redundancy check) and error correction codes (such as Hamming codes or Reed-Solomon codes) to detect and correct these errors.

MEMU operates on three error categories: Flash ECC errors, System RAM ECC errors and Peripheral RAM ECC errors. The MEMU distinguishes ECC events as: Correctable ECC (single-bit error), Uncorrectable ECC (multi-bit errors). When a memory error is detected, the MEMU can take various actions depending on the severity of the error and the system's configuration. These actions may include correcting the error automatically, notifying the operating system or application about the error, and even isolating faulty memory regions to prevent further errors. Some advanced MEMUs also provide features like error logging, error reporting, and error analysis. These features help in identifying the causes of memory errors, tracking error patterns, and facilitating system maintenance and troubleshooting.

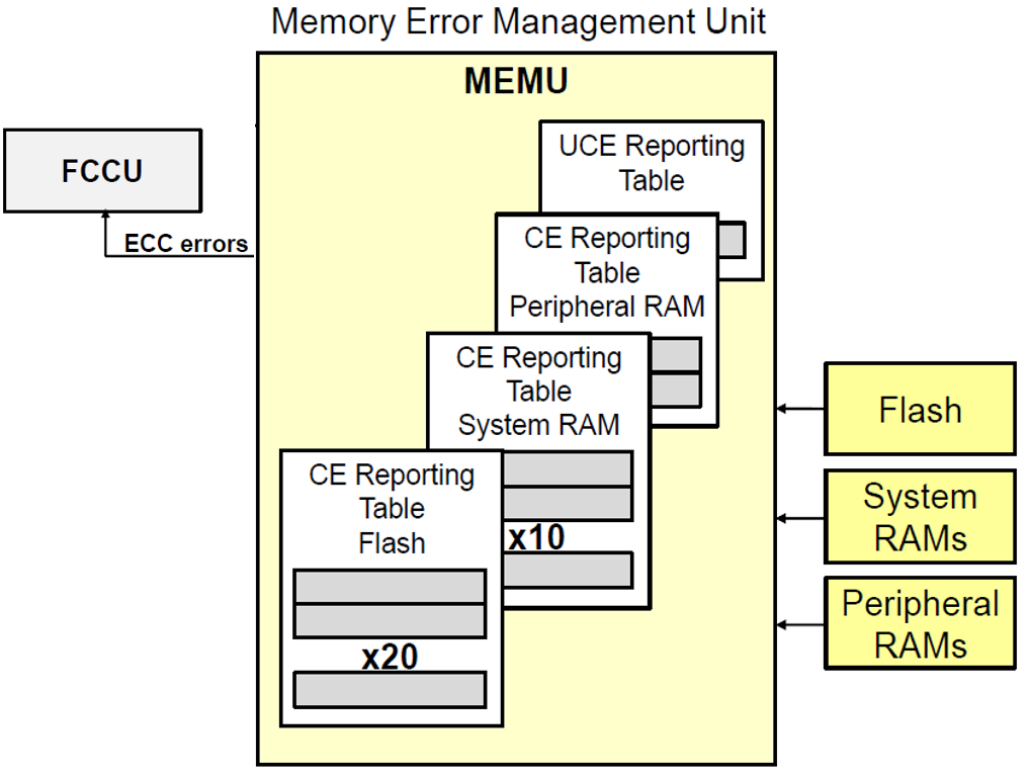

Figure 2.7: Block diagram of MEMU

2.6 Clock Monitoring Unit (CMU)

The CMU plays a crucial role in ensuring the integrity of the clock sources on the microcontroller. The CMUs supervise device clocks and detect issues like loss of lock, out-of-bounds frequency (over/under), and reference loss. If a monitored clock's frequency is lower than the reference clock or violates the frequency boundaries, the CMU detects and reports the issue to the Fault Collection and Control Unit (FCCU). The FCCU can then take appropriate corrective actions based on its configuration such as resetting the device, disable faulty clocks, or take other corrective measures, as necessary. In summary, It helps maintain the reliability and accuracy of clock sources and enables tasks such as calibration and time deviation calculation. CMUs use the 16 MHz Internal RC Oscillator (IRCOSC) to function independently from the monitored clocks.

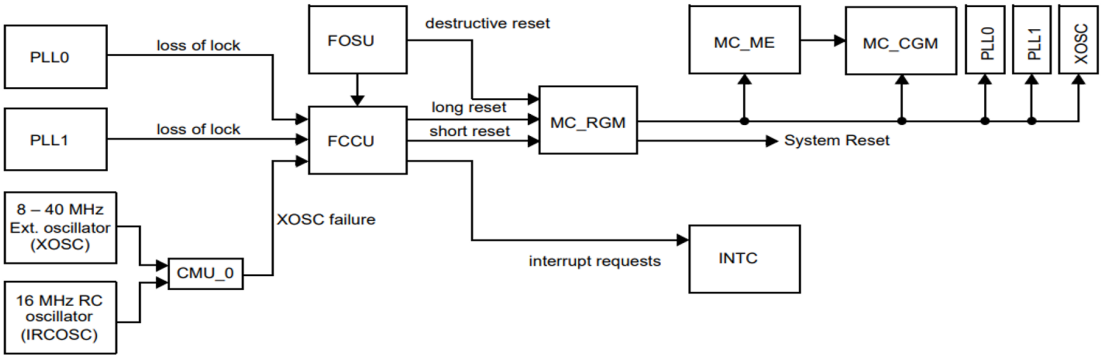

Figure 2.8: Block diagram of CMU

2.7 Power Management Unit (PMU)

The PMU is responsible for managing the power supply and consumption within the microcontroller, optimizing power usage, and providing various power-related features. Includes features such as

- **Power Monitoring and Supervision:** The PMU may include circuitry to monitor power-related parameters such as voltage, current, and temperature. It can provide these measurements to the microcontroller for monitoring purposes or to implement power protection mechanisms. For example, if the PMU detects an overvoltage or overcurrent condition, it can trigger appropriate actions to protect the microcontroller and connected devices.
- **Power Modes and Sleep Modes:** The PMU enables the microcontroller to operate in different power modes to conserve energy. These modes may include active mode, idle mode, low-power mode, and sleep modes with varying levels of power consumption and wake-up times.
- **Clock management:** Manages the clock signals that drive the MCU components.
- **Power-On/Reset Control:** The PMU controls the power-on and reset operations of the microcontroller.

2.8 Fault Collection and Control Unit (FCCU)

The Fault Collection and Control Unit (FCCU) plays a critical role in the SPC58 microcontroller. Serving as a hardware IP, the FCCU functions as a centralized capability for managing and gathering fault reports from various modules within the System-on-Chip (SoC). Importantly, the FCCU operates independently of the CPU, ensuring that a fault reporting mechanism remains available even in the event of CPU malfunctions. This specialized unit, designed with a focus on enhancing system and Electronic Control Unit (ECU) safety, significantly contributes to raising the overall level of system security and reliability.

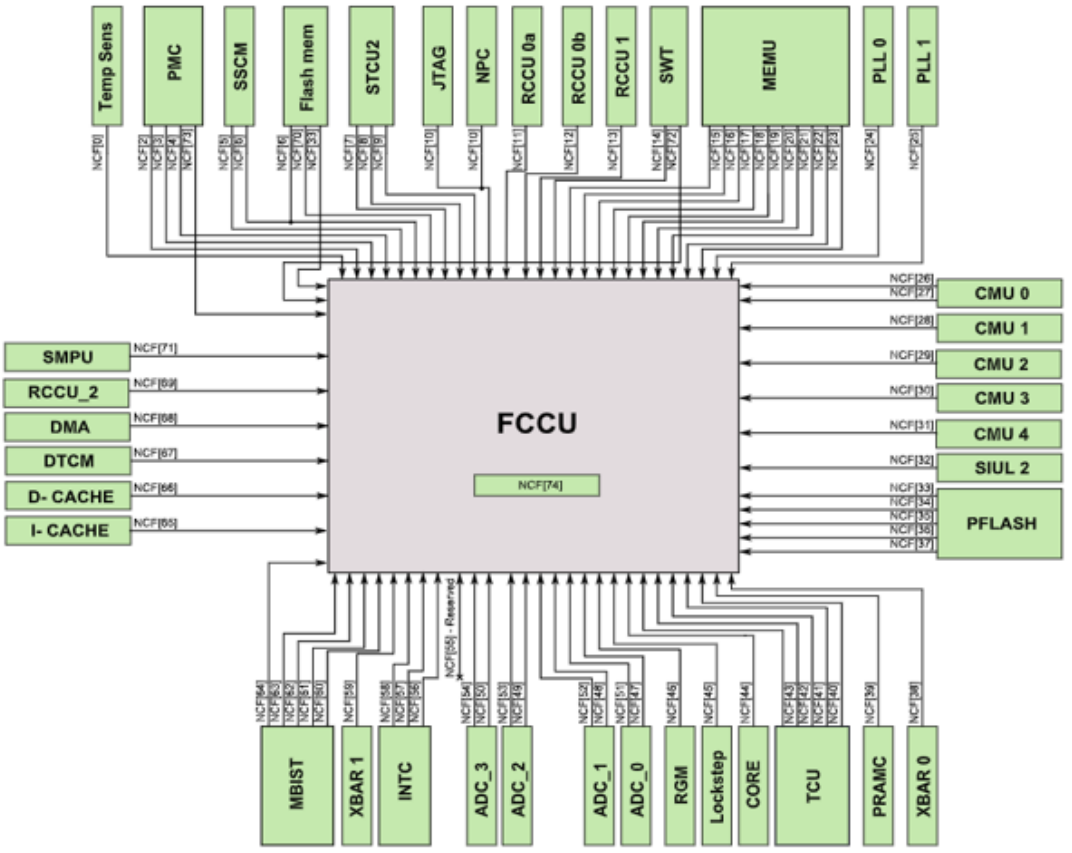

Figure 2.9: FCCU fault source mapping

2.8.1 FCCU Submodules

The FCCU consists of various submodules that collectively enable its functionality. Here's a brief explanation of these submodules:

- **REG intf:** This submodule includes the register file, which stores configuration settings and status information of the FCCU. It also contains the IPS (Internal Peripheral System) bus interface and IRQ (Interrupt Request) interface, which allow communication and synchronization with other components in the microcontroller.

- **HNSHK blocks:** The Handshake (HNSHK) blocks facilitate communication and synchronization between the REG intf submodule and the FSM (Finite State Machine) unit. As these submodules operate on different asynchronous clocks.
- **FSM unit:** The FSM unit is a significant submodule that implements the core functions of the FCCU. It manages fault detection, fault handling, and fault recovery procedures. Also includes the: Watchdog and Alarm timer.
- **FAULT intf:** The FAULT intf submodule is responsible for fault conditioning and management within the FCCU.
- **EOUTx units:** These submodules manage the EOUT (Error Output) interfaces. The EOUT interfaces are used to communicate fault-related information or status to external devices or systems.

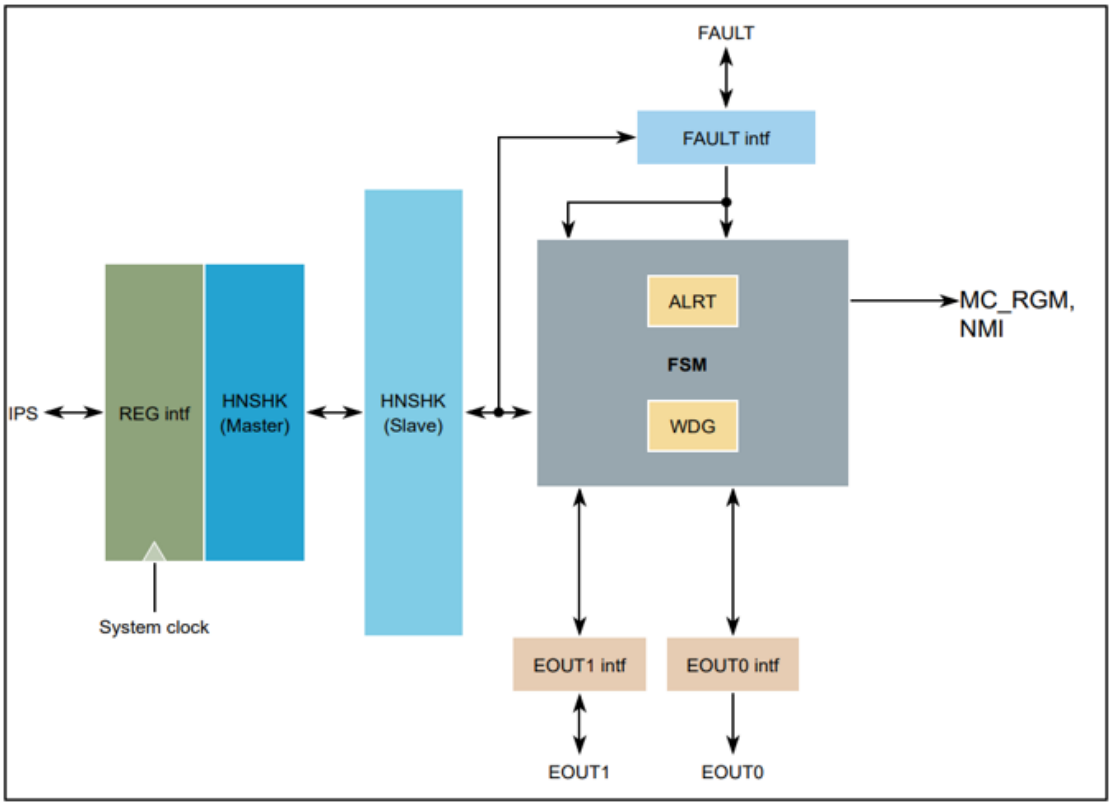

Figure 2.10: Block Diagram of FCCU

2.8.2 FCCU State Machine (FSM)

The FCCU (Fault Collection and Control Unit) incorporates a state machine that governs its operation and behavior. The state machine within the FCCU defines the different states and transitions that the unit can go through based on the occurrence of various events and conditions. It helps manage the fault detection, fault handling, and fault recovery processes within the microcontroller.

The FCCU state machine typically includes the following key states:

- **CONFIG:** The configuration state is used only to modify the default configuration of the FCCU.
- **NORMAL:** This is the FCCU's operating state when no faults are occurring. It is also the default state on the reset exit.
- **ALARM:** FCCU moves into the ALARM state when an unmasked recoverable fault occurs and the timeout is enabled.
- **FAULT:** When an unmasked recoverable fault occurs and is not recovered in given time.

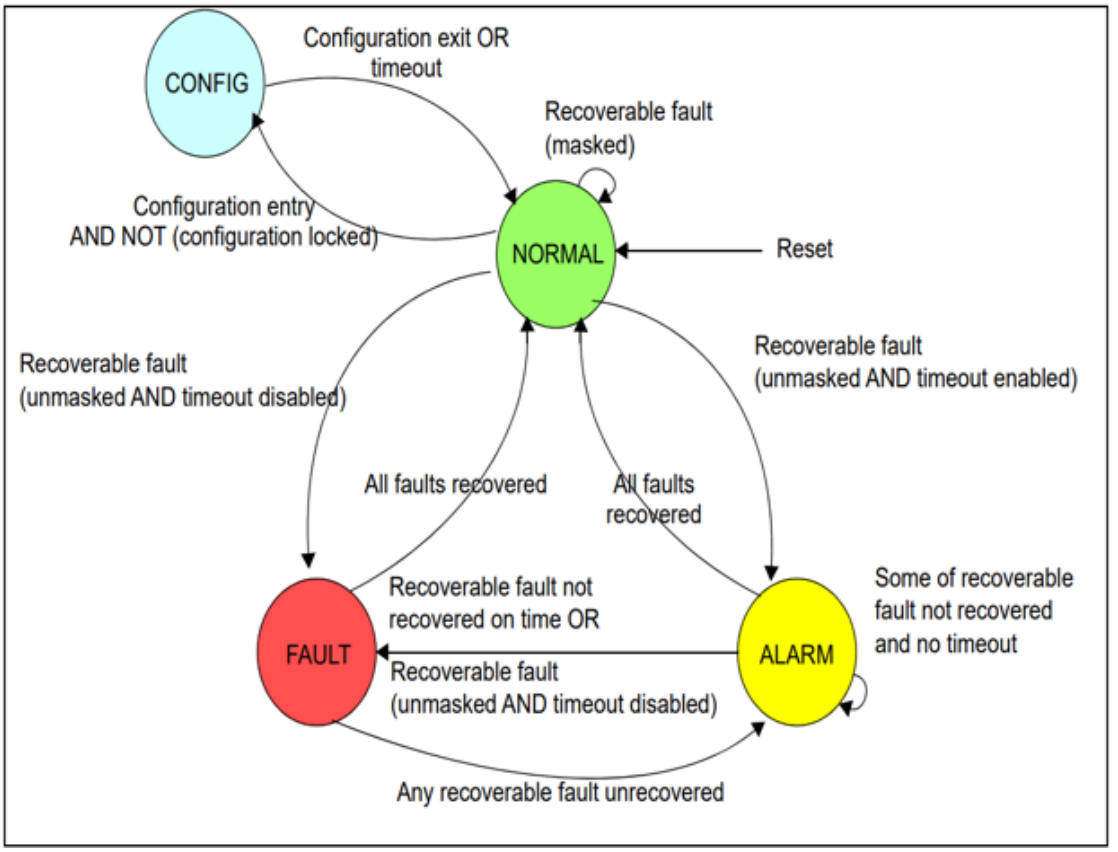

Figure 2.11: FCCU State Machine Diagram

2.9 Fault Description

In this section we will look at the fault descriptions related to the different faults and fault handling mechanisms studies in previous section and also run some similar safety tests on the hardware and study about the actions taken by safety module.

2.9.1 Temperature out of range 0/1

The device features temperature sensor modules located in different peripheral areas. A junction temperature sensor generates an output voltage proportional to the internal temperature. When temperature thresholds are exceeded, the FCCU is notified and takes appropriate actions. Additionally, the PMC can request a reset from the RGM module. These capabilities enable effective temperature monitoring and response, enhancing overall system safety.

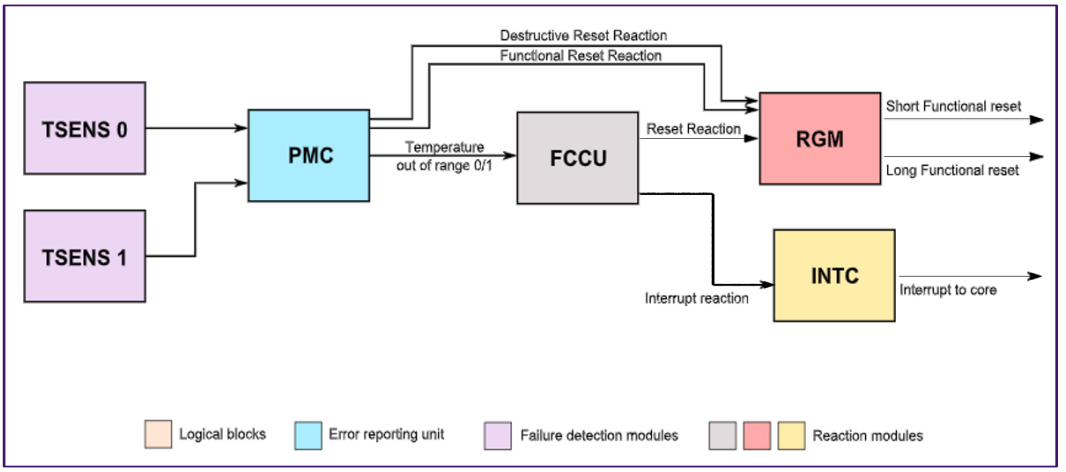

Figure 2.12: Temperature out of range 0/1

2.9.2 Low Voltage and High Voltage Detectors (LVD/HVD)

The PMC (Power Management Control unit) monitors supply signals and detects LVD and HVD events. If voltage thresholds are crossed, the PMC notifies the FCCU with a fault signal. In cases where the FCCU cannot respond, a direct path from the PMC ensures immediate action to prevent device damage. This mechanism is to prevents damage in critical voltage scenarios.

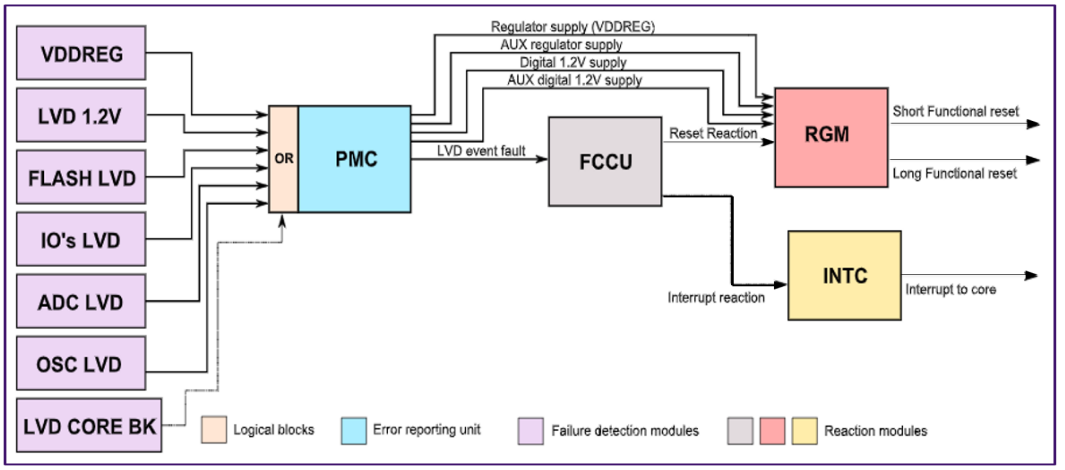

Figure 2.13: LVDs/HVDs ORed

2.9.3 STCU2 fault condition

The STCU2 fault condition arises when unexpected states occur in the LBIST/MBIST control signals during application runtime. This can inadvertently activate BIST mode for LBIST partitions or RAM arrays. If recoverable or unrecoverable faults are detected during self-tests, fault signals are forwarded to the FCCU for appropriate action.

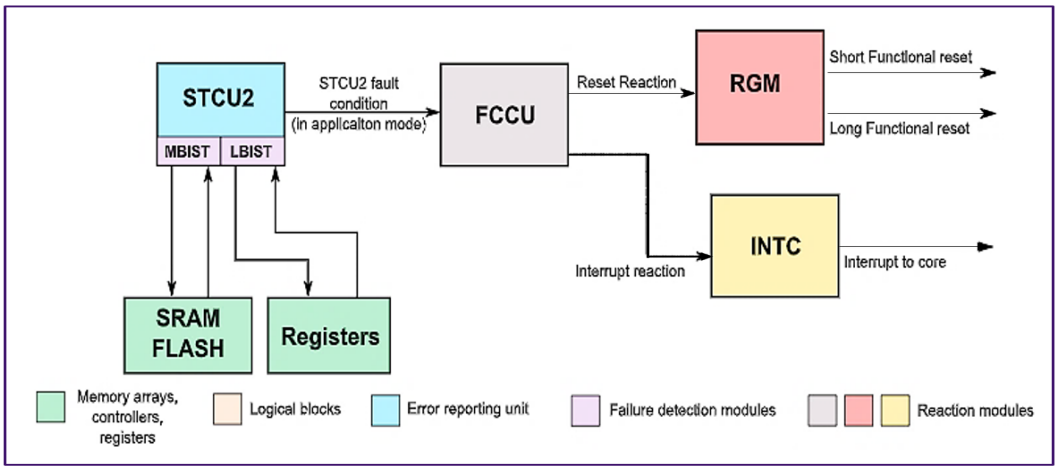

Figure 2.14: STCU2 fault condition

2.9.4 Core redundancy mismatch

The fault is triggered when redundancy checker unit (RCCU) identifies a mismatch between the output of the safety core and the corresponding output of the original core. Upon detection, the fault signal is transmitted to the FCCU for subsequent steps to be taken.

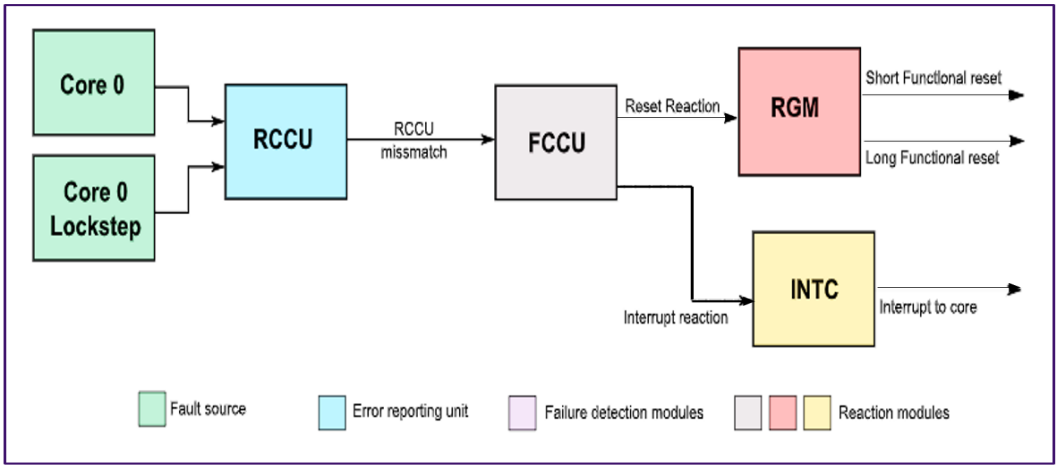

Figure 2.15: Core redundancy mismatch: Out of lockstep

2.9.5 System RAMs ECC error

The MEMU is responsible for managing and reporting error events related to the Error Correction Code (ECC) logic utilized in peripheral system RAM, SRAM, and flash memory. When an error event, whether correctable (single bit) or uncorrectable (multi-bit), occurs in the system RAMs, the MEMU receives an error signal. It proceeds to record the event, set the appropriate MEMU error flag, and subsequently reports this information to the FCCU.

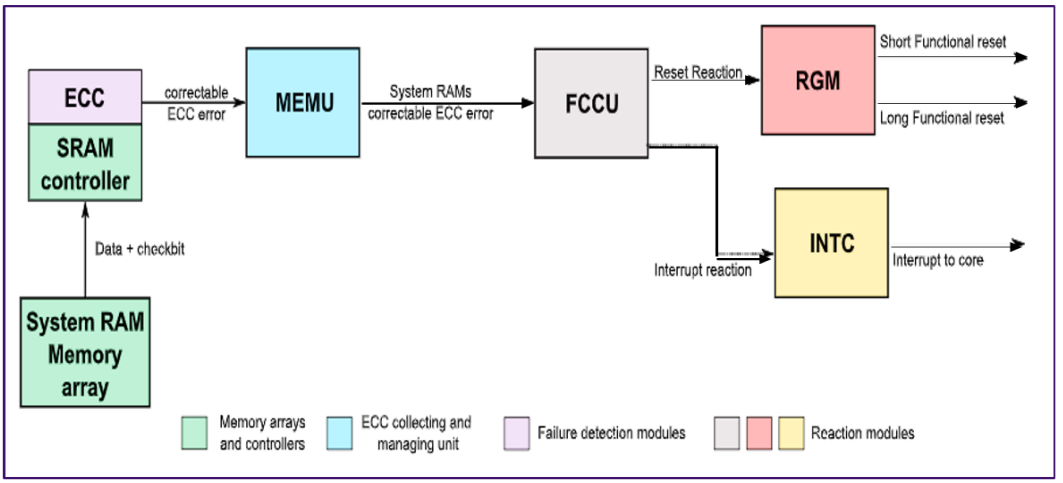

Figure 2.16: System RAMs ECC error

2.9.6 Clock frequency out of range

The CMUs are used to monitor issues like loss of lock, out-of-bounds frequency (over/under), and reference loss. Whenever such issues takes place, the CMU subsequently reports this information to the FCCU (Fault Collection and Control Unit).

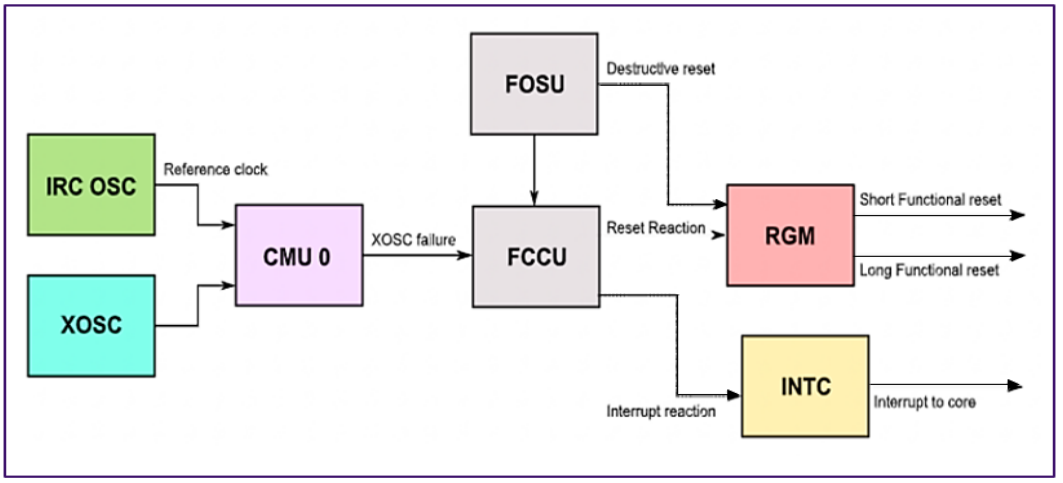

Figure 2.17: Clock frequency out of range

Chapter 3

Tools and Implementation

In this chapter, we will look at the software and hardware utilised in this project, as well as provide details on a few of the safety tests performed on the Infenion Device 4 40nm to gain a better understanding of the safety module discussed previously.

3.1 Software and Hardware

This section will look at the software and hardware that were used, as well as how they were integrated.

3.1.1 UDE (Universal Debug Engine)

The UDE (Universal Debug Engine) is an advanced tool designed for debugging, testing, and system analysis purposes. Offering a wide array of features, the UDE seamlessly integrates comprehensive capabilities for debugging, trace analysis, and runtime analysis, while maintaining an intuitive and efficient user interface. Key functionalities of the UDE include debugging support for C/C++ and assembler languages, real-time monitoring, system visualization, and system analysis. Additionally, the UDE extends its compatibility to encompass a diverse range of multicore System-on-Chips (SoCs) and microcontroller families.

3.1.2 UAD (Universal Access Device) 2 pro

The UAD2pro is a high-performance universal access device that provides debugging and trace capabilities for various microcontrollers and processors from different manufacturers. It serves as a bridge between a host computer and the target system, enabling software developers to analyze, debug, and optimize the code running on the target system. It offers fast communication with the target system, allowing for efficient debugging and analysis. The UAD2pro may support different communication interfaces such as JTAG, cJTAG, DAP, SWD, or others, depending on the specific version.

3.1.3 JTAG (Joint Test Action Group)

JTAG (Joint Test Action Group) is a standardized interface used for testing, debugging, and programming integrated circuits. It allows for debugging access, boundary scan testing, in-circuit emulation, and programming of devices. It provides a standardized connection and control mechanism for efficient and reliable testing and debugging of electronic systems.

3.1.4 Infenion Device 4 (TC389QP)

The second-generation AURIX microcontroller from Infineon is designed for automotive applications. It offers high performance with 6 TriCore cores running at 300 MHz, up to 16 MB flash memory, and 6.9 MB SRAM. It provides extensive connectivity options, robust security features, and complies with safety standards up to ASIL-D. The microcontroller supports over-the-air software updates, scalability, and easy migration from previous AURIX™ versions.

3.1.5 Hardware Integration

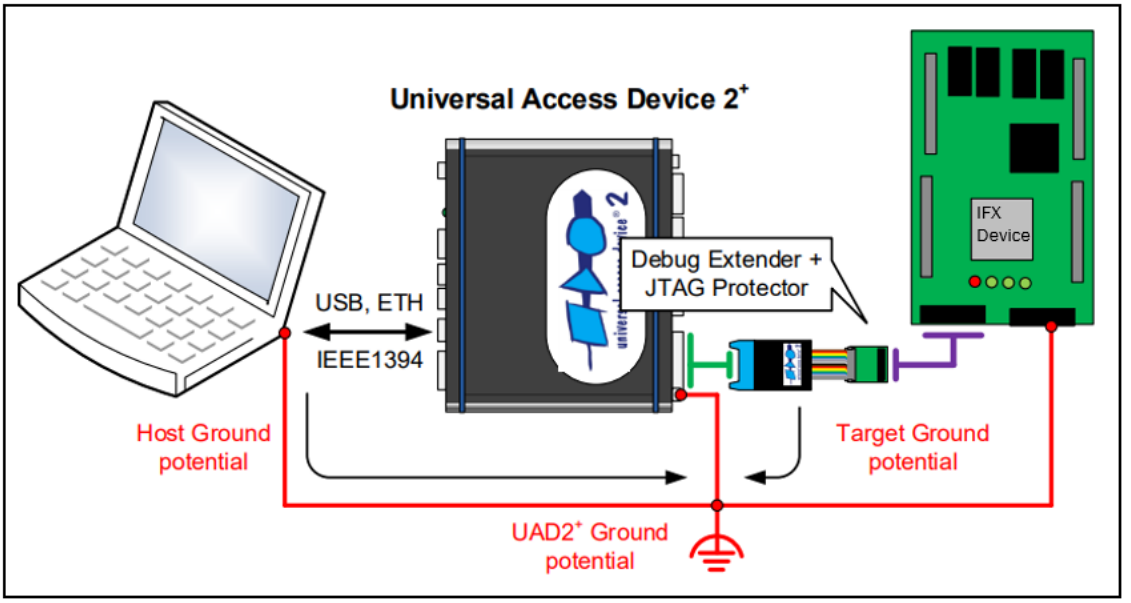

Figure 3.1: H/W Integration

3.2 Hardware Implementation

There are different safety tests that are available in the safety module of an Infenion device 4. Some of these test include Die Temperature Test, CPU Lockstep mode Test, EVR Test, Online BIST test etc... So here in this section we will be running few tests and discuss the results.

3.2.1 Die Temperature Test

This test is to monitor die temperature overflow and underflow with the help of Die Temperature Sensor (DTS) placed inside the Power Management System (PMS). Whenever there is an overflow or underflow of temperature detected by DTS the PMS is notified and error signal is forwarded to Safety Management Unit (SMU). The Registers used by PMS are PMS_DTSLIM and PMS_DTSSTAT.

DTSLIM (Die Temperature Sensor Limit Register)

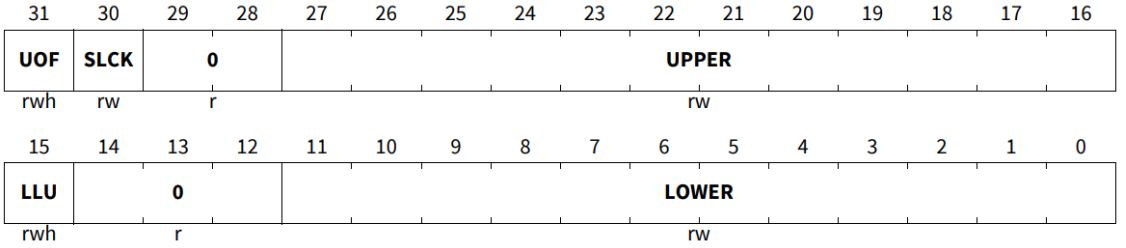

Table 3.1: DTSLIM Register Description

Field	Bits	Description
LOWER (Lower Limit)	11:0	This bit field defines the lower limit of the DTS temperature check. If result is less than or equal to the configured LOWER bitfield value; flag LLU is set.
LLU	15	0 = No temperature underflow was detected. 1 = A temperature underflow was detected.
UPPER (Upper Limit)	27:16	This bit field defines the upper limit of the DTS temperature check. If the measurement result is greater than or equal to the configured UPPER bitfield value; flag UOF is set.
UOF	31	0 = No temperature overflow was detected. 1 = A temperature overflow was detected.

DTSSTAT (Die Temperature Sensor Status Register)

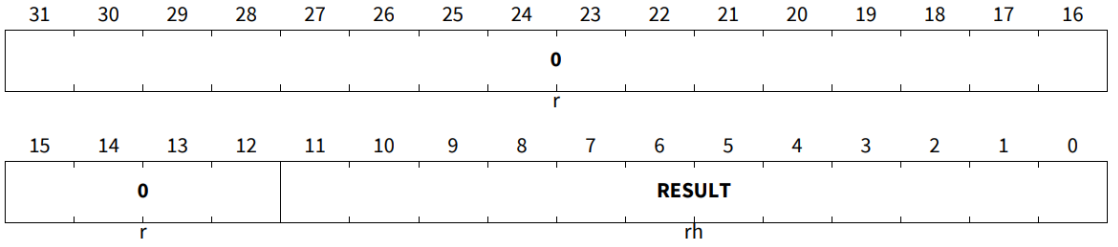

Table 3.2: DTSSTAT Register Description

Field	Bits	Description
RESULT	11:0	This bit field shows the result of the DTS measurement. The value given is directly related to the die temperature.
0	31:12	Reserved.

Before Fault Injection

Lower Limit = 0x6D6.
Upper Limit = 0xFFF.
Current Temp Status (DTSSTAT) = 0x91A.
We can see that Current Temp is within the range of Upper and Lower Limit. Therefore, No UOF (overflow) or LLU(underflow) flag bit is set.

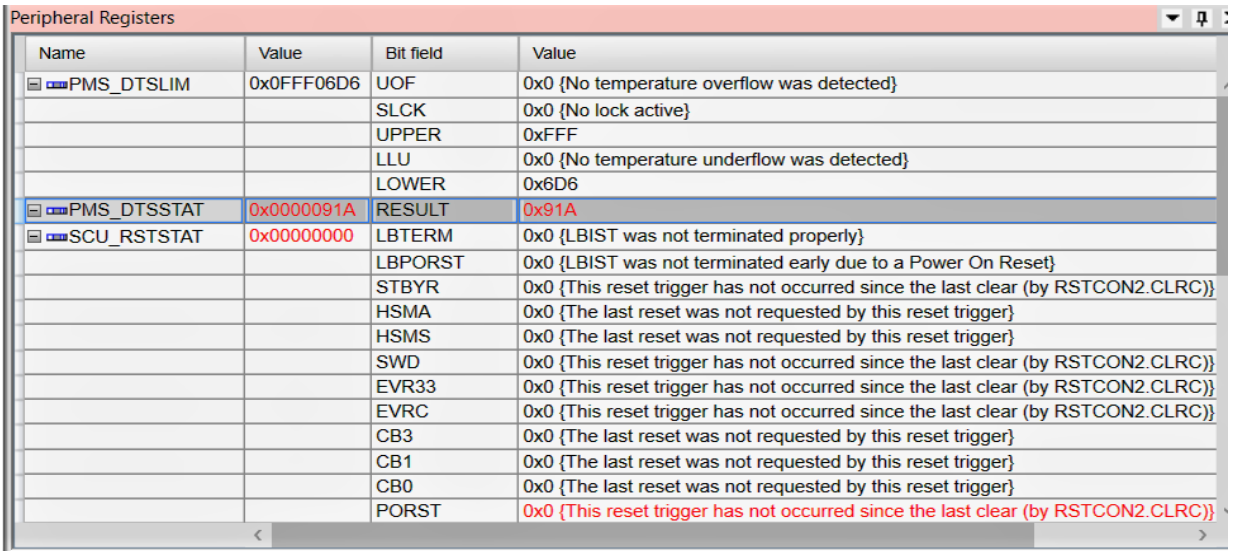

Figure 3.2: Before Fault Injection

After Fault Injection

Lower Limit = 0xFFF.
Upper Limit = 0x000.
Current Temp Status (DTSSTAT) = 0x91A.
By changing Upper and Lower Limit values respectively we can see that the Current temp crosses the given range. Therefore, UOF(overflow) and LLU(underflow) flag bit are set respectively. We can also detect a rise in SMU alarms. SMU Alarm Group 9 (SMU_AG9) SF1 (underflow) and SF0 (overflow) alarms.

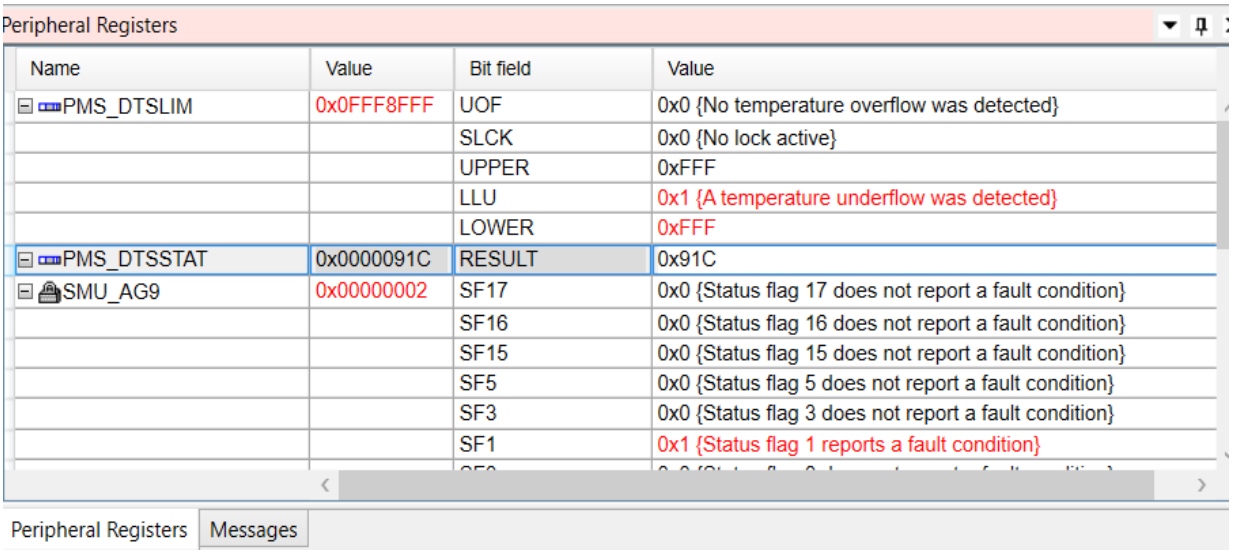

Figure 3.3: Underflow Fault Injection

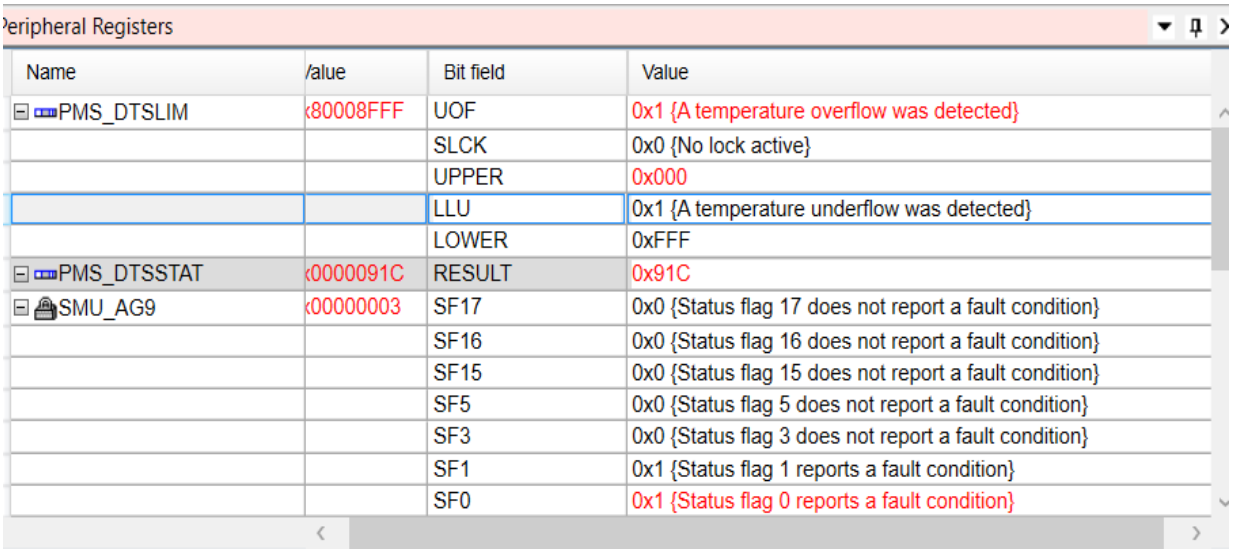

Figure 3.4: Overflow Fault Injection

Chapter 4

Conclusions and Future Scope

This chapter summarises the project's findings as well as the accomplishments made as a result of it. The acquired results are reviewed in detail, as well as future work that may be done to increase the quality of the summary.

4.1 Conclusion

In conclusion, this paper presents the design and implementation of a functionally safe automotive System-on-Chip (SoC). The proposed SoC architecture integrates robust safety mechanisms, adheres to industry standards such as ISO 26262. The research demonstrates the effectiveness of safety features including fault detection, error correction, redundancy, and comprehensive self-diagnostic capabilities.

It also presents some of the safety tests done on the hardware (Infineon Device 4) and discusses the registers and alarms generated by the safety modules. It shows how users can access detailed information about the hardware's safety tests by using the UDE. This includes information about the test methodology, results, and any produced alarms. The UDE simplifies the analysis of registers and alarms generated by the device's safety modules.

4.2 Future Scope

The paper lays the foundation for a functionally safe automotive SoC design. Further research can explore the development of more advanced safety mechanisms, such as fault prediction and proactive fault mitigation techniques. This could involve leveraging machine learning and artificial intelligence algorithms to analyze sensor data and predict potential faults before they occur.

As vehicles become increasingly connected and autonomous, ensuring not only functional safety but also cybersecurity becomes crucial. Future studies can focus on integrating robust security features into the SoC design to protect against cyber threats and unauthorized access, ensuring the overall safety and integrity of automotive systems.

References

[1] EDN. Approaches to functional safety in automotive design. https://www.edn.com/approaches-to-functional-safety-in-automotive-design/.

[2] Karl Greb and Dev Pradhan. Hercules™ microcontrollers: Real-time mcus for safety-critical products. *white Paper*, 2011.

[3] Eric C Jones and Andrew J Allan. Mbist device for use with ecc-protected memories, February 28 2017. US Patent 9,583,216.

[4] Jean-Philippe Meunier. Automotive functional safety: The evolution of fail safe to fail operational architecture. https://www.nxp.com/company/blog/.

[5] Alessandra Nardi and Antonino Armato. Functional safety methodologies for automotive applications. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 970–975. IEEE, 2017.

[6] Rob Palin, David Ward, Ibrahim Habli, and Roger Rivett. Iso 26262 safety cases: Compliance and assurance. 2011.

[7] Freescale Semiconductors. Functional safety implementations in modern mcus. https://www.eetimes.com/functional-safety-implementations-in-modern-mcus/.

[8] NXP Semiconductors. Using the built-in self-test (bist) on the mpc5744p.

[9] Shinichi Shibahara. Functional safety soc for autonomous driving. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–8. IEEE, 2018.

[10] Joseph Yiu. Design of soc for high reliability systems with embedded processors. In *Embedded World Conference*, 2015.

internship

ORIGINALITY REPORT

14%	13%	4%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.st.com Internet Source	3%
2	www.infineon.com Internet Source	2%
3	www.coursehero.com Internet Source	2%
4	archive.eetindia.co.in Internet Source	<1%
5	Shinichi Shibahara, Chikafumi Takahashi, Kazuki Fukuoka, Yuko Kitaji et al. "A 16 nm FinFET Heterogeneous Nona-Core SoC Supporting ISO26262 ASIL B Standard", IEEE Journal of Solid-State Circuits, 2017 Publication	<1%
6	utpedia.utp.edu.my Internet Source	<1%
7	www.nxp.com Internet Source	<1%
8	www.edn.com Internet Source	

		<1 %
9	www-blue.infineon.com Internet Source	<1 %
10	www.kscst.iisc.ernet.in Internet Source	<1 %
11	www.freepatentsonline.com Internet Source	<1 %
12	socrates.vsau.org Internet Source	<1 %
13	ebin.pub Internet Source	<1 %
14	docplayer.net Internet Source	<1 %
15	pure.tue.nl Internet Source	<1 %
16	Jean-Claude Geffroy, Gilles Motet. "Design of Dependable Computing Systems", Springer Nature, 2002 Publication	<1 %
17	repository.umy.ac.id Internet Source	<1 %
18	The Handbook of Data Communications and Networks, 2004. Publication	<1 %

19	www.cse.ust.hk Internet Source	<1 %
20	edoc.pub Internet Source	<1 %
21	www.scribd.com Internet Source	<1 %
22	"Recent Findings in Intelligent Computing Techniques", Springer Science and Business Media LLC, 2018 Publication	<1 %
23	S. Banerjee, D.R. Chowdhury, B.B. Bhattacharya. "A Programmable Built-In Self-Test for Embedded DRAMs", 2005 IEEE International Workshop on Memory Technology, Design, and Testing (MTDT'05), 2005 Publication	<1 %
24	archive.org Internet Source	<1 %
25	www.esi.nl Internet Source	<1 %
26	Chen, M., and J. Lu. "High-precision Motion Control for a Linear Permanent Magnet Iron Core Synchronous Motor Drive in Position Platform", IEEE Transactions on Industrial Informatics, 2013. Publication	<1 %

27	"An Overview of BIST", Frontiers in Electronic Testing, 2002 Publication	<1 %
28	Marc Osajda. "Solutions for Safety Critical Automotive Applications", Advanced Microsystems for Automotive Applications 2010, 2010 Publication	<1 %
29	Shubham Salvi, Arunava Mitra, Shubham Devlekar, Archana Choudhary, Shital Patil, Surendra Bhosale. "Novel ECU Design Architecture using DSP and FPGA for Electric Vehicle", 2022 IEEE 10th Power India International Conference (PIICON), 2022 Publication	<1 %
30	cache.freescale.com Internet Source	<1 %
31	www.slideshare.net Internet Source	<1 %
32	"Proceedings", 2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2017 Publication	<1 %
33	saemobilus.sae.org Internet Source	<1 %
34	www.arxiv-vanity.com Internet Source	

<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		